DNNrocket Paging

# Introduction

DNNrocket has a standard paging code section. This included page of Simplsity and part Server Code.

Data for controlling the paging is saved in the HeaderData property of Simplisity.SimplisityRazor and past to the razor template by that.

It also uses cookies to keep track of changing pagesize and page.

Different implementation can be done, but here are a couple of examples.

# Static Page Size

The is the most simply implementation. Using a static page size the user cannot change.

var rowcount = objCtrl.GetListCount(HeaderInfo.PortalId, -1, "PAY", filter, "", \_systemid);

\_searchInfo.SetXmlProperty("genxml/hidden/rowcount", rowcount.ToString()); // paging template needs this

\_searchInfo.SetXmlProperty("genxml/hidden/pagesize", "2"); // static pagesize.

var pagesize = \_searchInfo.GetXmlPropertyInt("genxml/hidden/pagesize");

var page = \_searchInfo.GetXmlPropertyInt("genxml/hidden/page");

if (page == 0) page = 1;

\_dataList = objCtrl.GetList(-1, \_moduleid, "PAY", filter, \_langRequired, " order by R1.ItemId DESC", 0, page, pagesize, rowcount, \_systemid);

SaveHeader(\_searchInfo);

The \_searchInfo has been posted back as search data and then used to get the data list. After the \_searchInfo is saved as the header. This code is in a list population function, so it is called even if no search data is passed to the server, therefore paging data is always passed to the razor template.

# The Paging Template

The standard template to display paging is:

@RenderTemplate("Paging.cshtml", "\\DesktopModules\\DNNrocket\\api", "config-w3", Model)

This uses the simplicity class “simplicity\_pageclick”

$('.simplisity\_pageclick').each(function (index) {

    $(this).attr("s-index", index);

    $(this).unbind("click");

    $(this).click(function () {

        var p = $(this).attr("s-page");

        simplisity\_setCookieValue('s-page', p);

        simplisity\_pagechange(this, cmdurl);

    });

});

This sets the s-page cookie, so we know which page has been selected, it then calls the “simplicity\_pagechange” function

function simplisity\_pagechange(element, cmdurl) {

    var scmd = $(element).attr('s-cmd');

    if (typeof scmd !== 'undefined' && scmd !== '') {

        simplisity\_callserver(element, cmdurl);

    } else {

        location.reload();

    }

}

Here we decided is the paging has a Simplsity command in the page click, if not the page is reloaded and the cookie “s-page” is used to get the page that needs to be displayed.

The page number is passed to the server by Simplisity, by adding to to the json that is passed to the server.

// Add paging data

        var pagesize = simplisity\_getCookieValue('s-pagesize');

if (typeof pagesize !== 'undefined') {

    jsonDataF['pagesize'] = simplisity\_encode(pagesize) || '';

}

var page = simplisity\_getCookieValue('s-page');

if (typeof page !== 'undefined') {

    jsonDataF['page'] = simplisity\_encode(page) || '';

}

The paging data it then picked up on the server by looking at “genxml/hidden/page” and “genxml/hidden/pagesize” in the posted SimplisityInfo class.

## Turn on Ajax Paging

The paging template allows for Ajax. This is done by adding the commands and required data to the HeaderData. It is then used by the template to add the data to the simplsity\_pageclick.

<a class="w3-button w3-hover-theme simplisity\_pageclick" s-page="1" s-cmd="@info.GetXmlProperty("genxml/s-paging-cmd")" s-post="@info.GetXmlProperty("genxml/s-paging-post")" s-fields="@info.GetXmlProperty("genxml/s-paging-fields")" s-return="@info.GetXmlProperty("genxml/s-paging-return")" href="#">|❮</a>

Server-Side code. \_searchInfo is returned as the HeaderData.

// setup ajax call for paging

\_searchInfo.SetXmlProperty("genxml/s-paging-cmd", "rocketpayform\_search");

\_searchInfo.SetXmlProperty("genxml/s-paging-post", "#headerdata");

\_searchInfo.SetXmlProperty("genxml/s-paging-fields", "tabid:" + \_tabid + ",moduleid:" + \_moduleid + ",selecteditemid:,interfacekey:rocketpayform");

\_searchInfo.SetXmlProperty("genxml/s-paging-return", "#simplisity\_startpanel");

# Dynamic User Page Size

The dynamic page size is driven by the “simplicity\_pagesize” class. This class can be placed on a dropdownlist to allow the user to select a pagesize.

@DropDownList(headerData, "genxml/hidden/pagesize", "5,10,20,50,100", "5,10,20,50,100", "class='w3-input w3-border simplisity\_pagesize ' title='Page Size'", "20")

$('.simplisity\_pagesize').each(function (index) {

    if (simplisity\_isSelect(this)) {

        $(this).unbind("change");

        $(this).change(function () {

            simplisity\_setCookieValue('s-pagesize', $(this).val());

            simplisity\_pagechange(this, cmdurl);

        });

    }

    if (simplisity\_isTextInput(this)) {

        $(this).unbind("click");

        $(this).click(function () {

            simplisity\_setCookieValue('s-pagesize', $(this).val());

            simplisity\_pagechange(this, cmdurl);

        });

    }

});

This will set the pagesize cookie and call “simplicity\_pagechange” function. As we’ve seen this will trigger the post to the server as “genxml/hidden/pagesize” **and can be then used to set the pagesize** for the data list. Ajax call also be placed onto the dropdownlist by adding the required Simplisity classes and data.